Java Programming

Classes

Inheritance

Arash Habibi Lashkari

Ph.D. Candidate of UTM University

Kuala Lumpur, Malaysia

CONSTRUCTOR METHODS

- Consider the following class definition:

```
class Account {
    private String acctNo;
    private boolean active;
    private String owner;

    public void displayInfo( ) {
        System.out.println("Acct number: "+acctNo);
        System.out.println("Owner: "+owner);
        System.out.print("Status: ");
        if (!active)
            System.out.print("NOT ");
        System.out.println("ACTIVE");
    }
}
```

CONSTRUCTOR METHODS

- What happen when the following statements are executed?

```
Account acct = new Account( );
acct.displayInfo( );
```

- When an object is created, its attributes should be initialized. Example:

```
Account acct = new Account( );
acct.initialize("010-99333-03", "Nada Asyiqin",
                true);
acct.displayInfo( );
```

Problem: It is easy to forget to do the initialization.

CONSTRUCTOR METHODS

- Can object initialization be done during object creation?
- Two common ways of initializing attributes during object creation:
  - Using instance variable initializers
  - Using constructor methods

CONSTRUCTOR METHODS

Method 1: Instance Variable Initializer

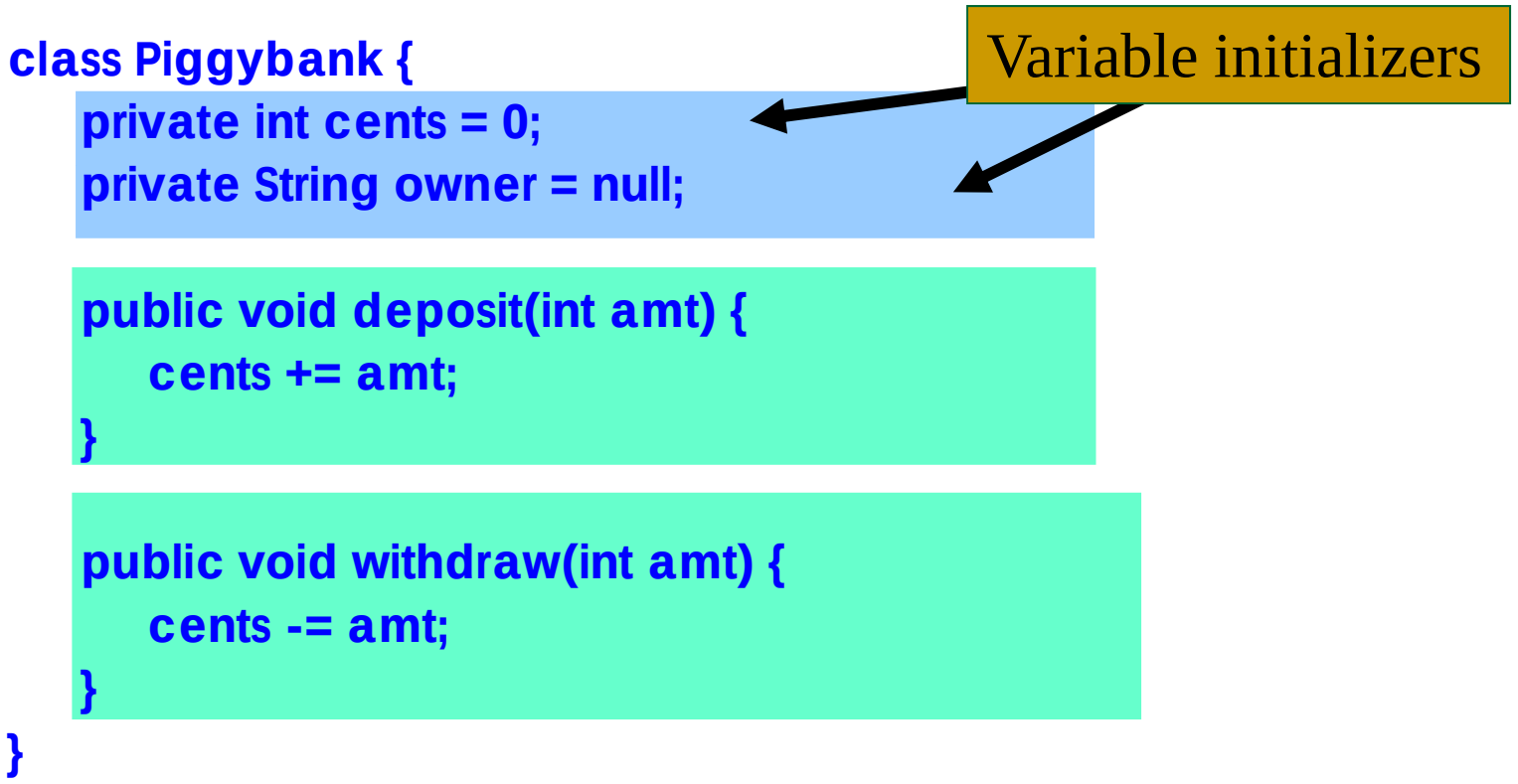
```
class Piggybank {
    private int cents = 0;
    private String owner = null;

    public void deposit(int amt) {
        cents += amt;
    }

    public void withdraw(int amt) {
        cents -= amt;
    }
}
```

CONSTRUCTOR METHODS

Method 1: Instance Variable Initializer



CONSTRUCTOR METHODS

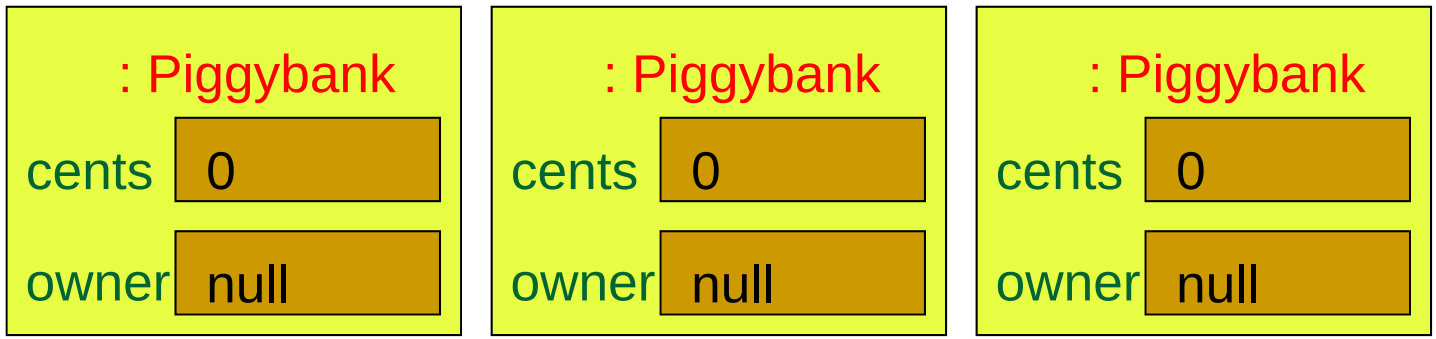
- For each *Piggybank* object created, its instance variable *cents* will be initialized to 0.

```
for (int i=0; i < 3; i++)
    new Piggybank( );
```

CONSTRUCTOR METHODS

- For each *Piggybank* object created, its instance variable *cents* will be initialized to 0.

```
for (int i=0; i < 3; i++)
    new Piggybank( );
```



CONSTRUCTOR METHODS

Method 2: Constructor Methods

```
class Piggybank {
    private int cents;
    private String owner;

    public Piggybank( ) {
        cents = 0;
        owner = null;
    }
    public void deposit(int amt) {
        cents += amt;
    }
    public void withdraw(int amt) {
        cents -= amt;
    }
}
```

CONSTRUCTOR METHODS

Method 2: Constructor Methods

```
class Piggybank {
    private int cents;
    private String owner;

    public Piggybank( ) {
        cents = 0;
        owner = null;
    }
    public void deposit(int amt) {
        cents += amt;
    }
    public void withdraw(int amt) {
        cents -= amt;
    }
}
```

Constructor method

CONSTRUCTOR METHODS

- The constructor method is executed each time a Piggybank object is created. When executed,
  - the *cents* attribute of the created object is initialized to 0.
  - the *owner* attribute is set to *null*.

CONSTRUCTOR METHODS

- Now consider the following code:

```
for (int i=0; i < 3; i++)
    new Piggybank( );
```

CONSTRUCTOR METHODS

- Now consider the following code:

```
for (int i=0; i < 3; i++)
    new Piggybank( );
```

CONSTRUCTOR METHODS

- Now consider the following code:

```
for (int i=0; i < 3; i++)
    new Piggybank( );
```

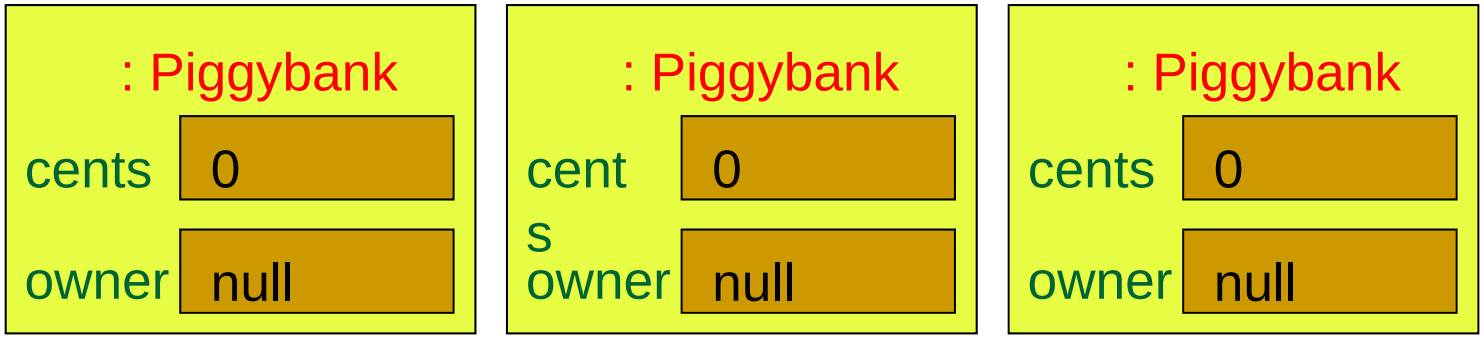
```
public Piggybank( ) {
    cents = 0;
    owner = null;
}
```

CONSTRUCTOR METHODS

- Now consider the following code:

```
for (int i=0; i < 3; i++)
    new Piggybank( );
```

```
public Piggybank( ) {
    cents = 0;
    owner = null;
}
```



CONSTRUCTOR METHODS

- The constructor method is a suitable place for putting in the necessary code for initializing an object when it is being created.
- Constructors are not allowed to return any value at the end of their execution.

  In fact, constructors do not have a return type.

CONSTRUCTOR METHODS

- The constructor method is a suitable place for putting in the necessary code for initializing an object when it is being created.
- Constructors are not allowed to return any value at the end of their execution.

  In fact, constructors do not have a return type.

```
public Piggybank( ) {
    cents = 0;
    owner = null;
}
```

CONSTRUCTOR METHODS

- Another restriction: the name of a constructor must be the same as the name of the class.

```
class Piggybank {
    ...
    public Piggybank( )
    {
        cents = 0;
        owner = null;
    }
}
```

CONSTRUCTOR METHODS

- Another restriction: the name of a constructor must be the same as the name of the class.

```
class Piggybank {

    public Piggybank( )
    {
        cents = 0;
        owner = null;
    }
}
```

CONSTRUCTOR METHODS

- Constructors can be parameterized. For an object to execute a constructor with parameters, the required parameters need to be passed when that object is created.

CONSTRUCTOR METHODS

```
class Piggybank {
    private int cents;
    private String owner;

    public Piggybank(int amt, String name) {
        cents = amt;
        owner = name;
    }
    public void deposit(int amt) {
        cents += amt;
    }
    public void withdraw(int amt) {
        cents -= amt;
    }
}
```

CONSTRUCTOR METHODS

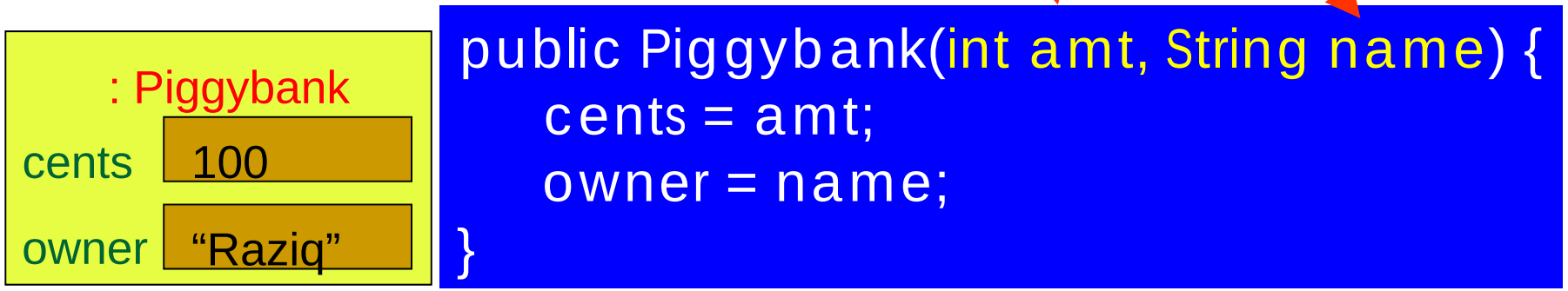
- An example of creating a Piggybank object:

```
Piggybank pb = new
                Piggybank(100,“Raziq”);
```

CONSTRUCTOR METHODS

- An example of creating a Piggybank object:

```
Piggybank pb = new
                Piggybank(100,"Raziq");
```



CONSTRUCTOR METHODS

- Each class must define at least one constructor method.
- If no constructors are defined for a class, the Java compiler will automatically insert a default constructor.
- The default constructor inserted will contain no parameter and has an empty body.

CONSTRUCTOR METHODS

```
class Account {
   private String acctNo;
   private boolean active;
   private String owner;
   public Account(String nbr, String name) {
        acctNo = nbr;
        owner = name;
        active = false;
   }
   public void displayInfo( ) {
        System.out.println(“Account number: "+acctNo);
        System.out.println(“Owner: "+owner);
        System.out.print("Status: ");
        if (!active)
           System.out.print(“NOT ");
        System.out.println("ACTIVE");
   }
}
```

CONSTRUCTOR METHODS

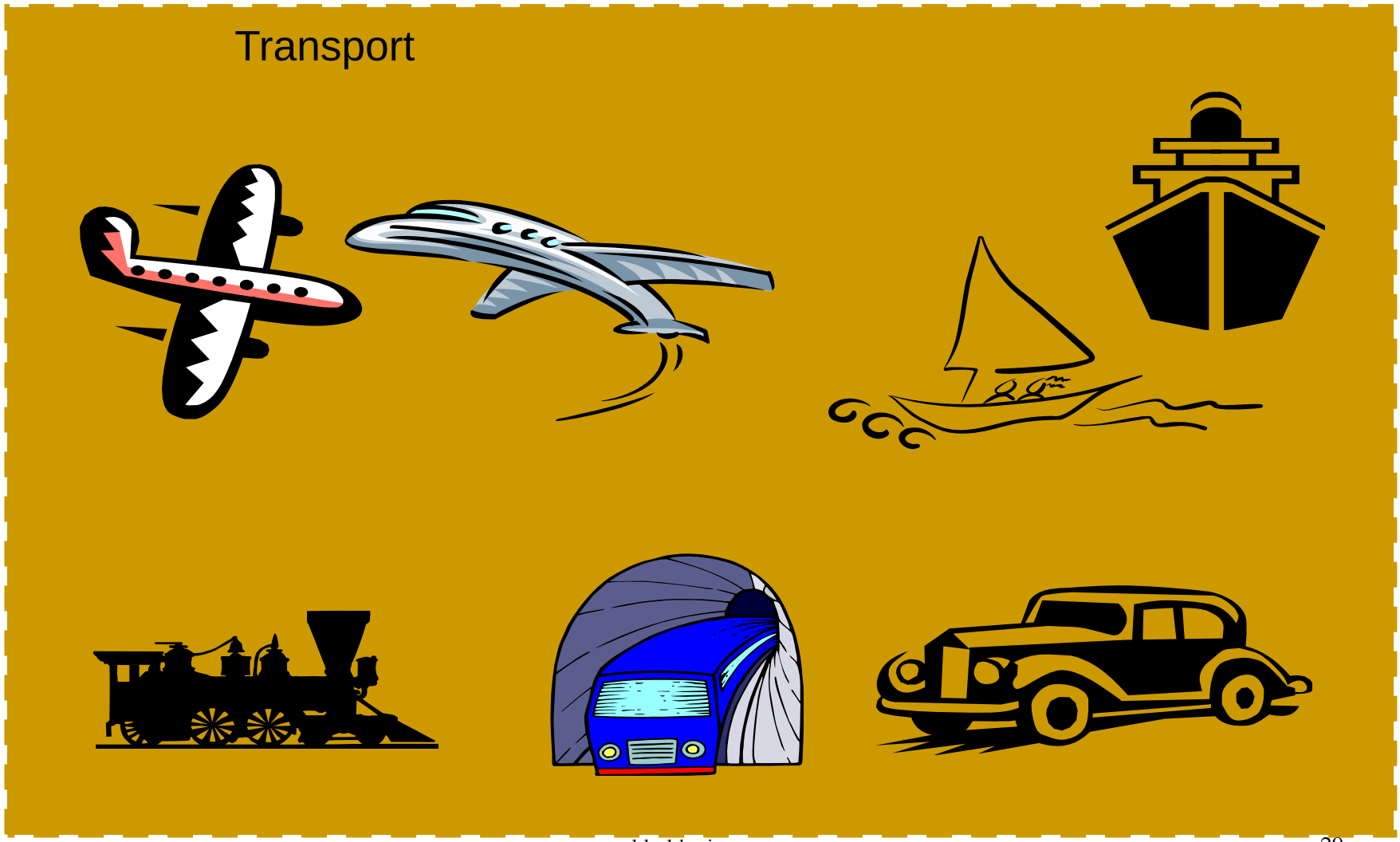
```
class Application {
    public static void main(String[ ] args) {
        Account acct;
        acct = new Account("010-99333-    03", "Nada Asyiqin");
        acct.displayInfo( );
    }
}
```

Inheritance

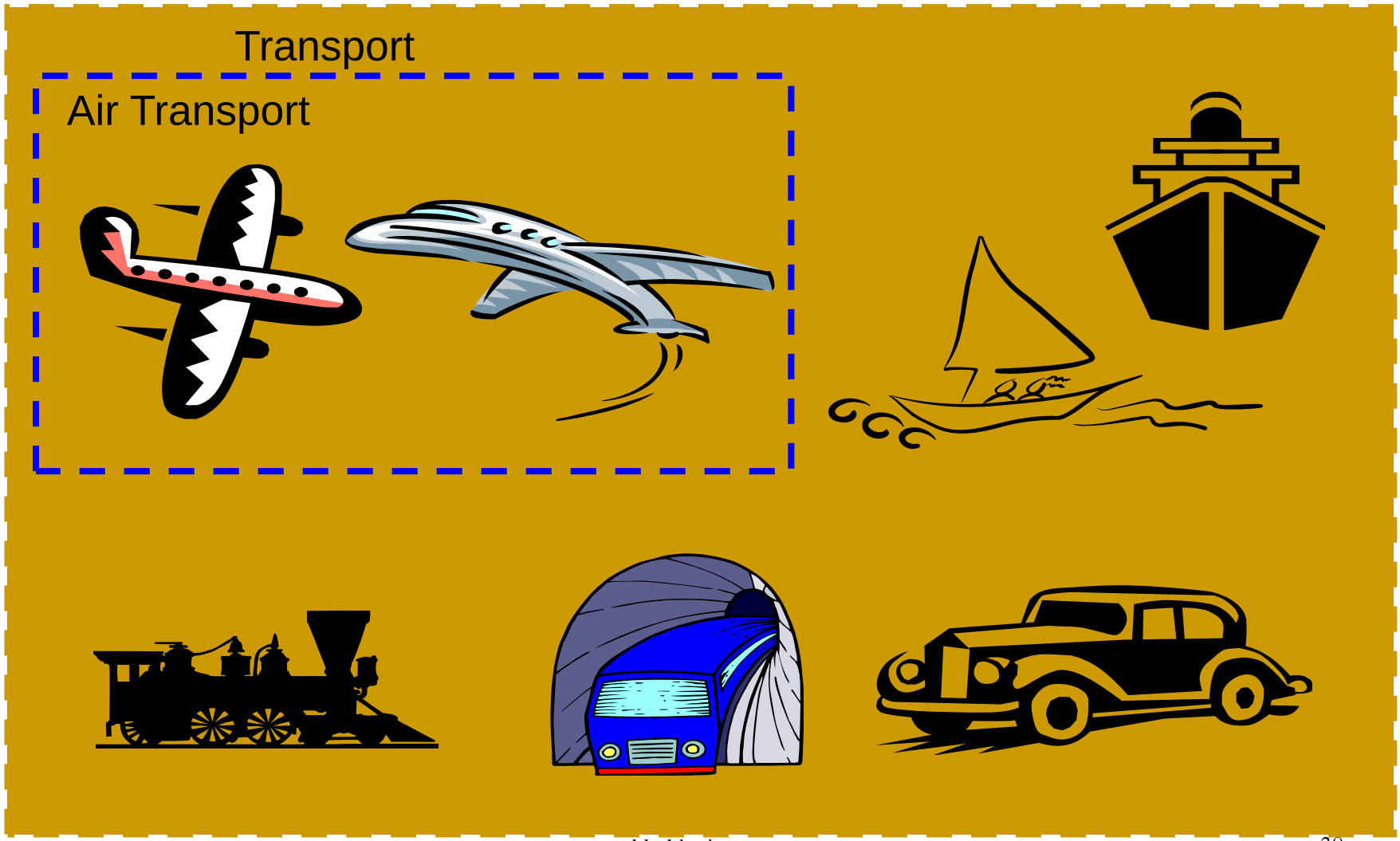
Inheritance Relationship

- The inheritance relationship can be viewed as a relationship between an object category and its subcategories.
  - For example...

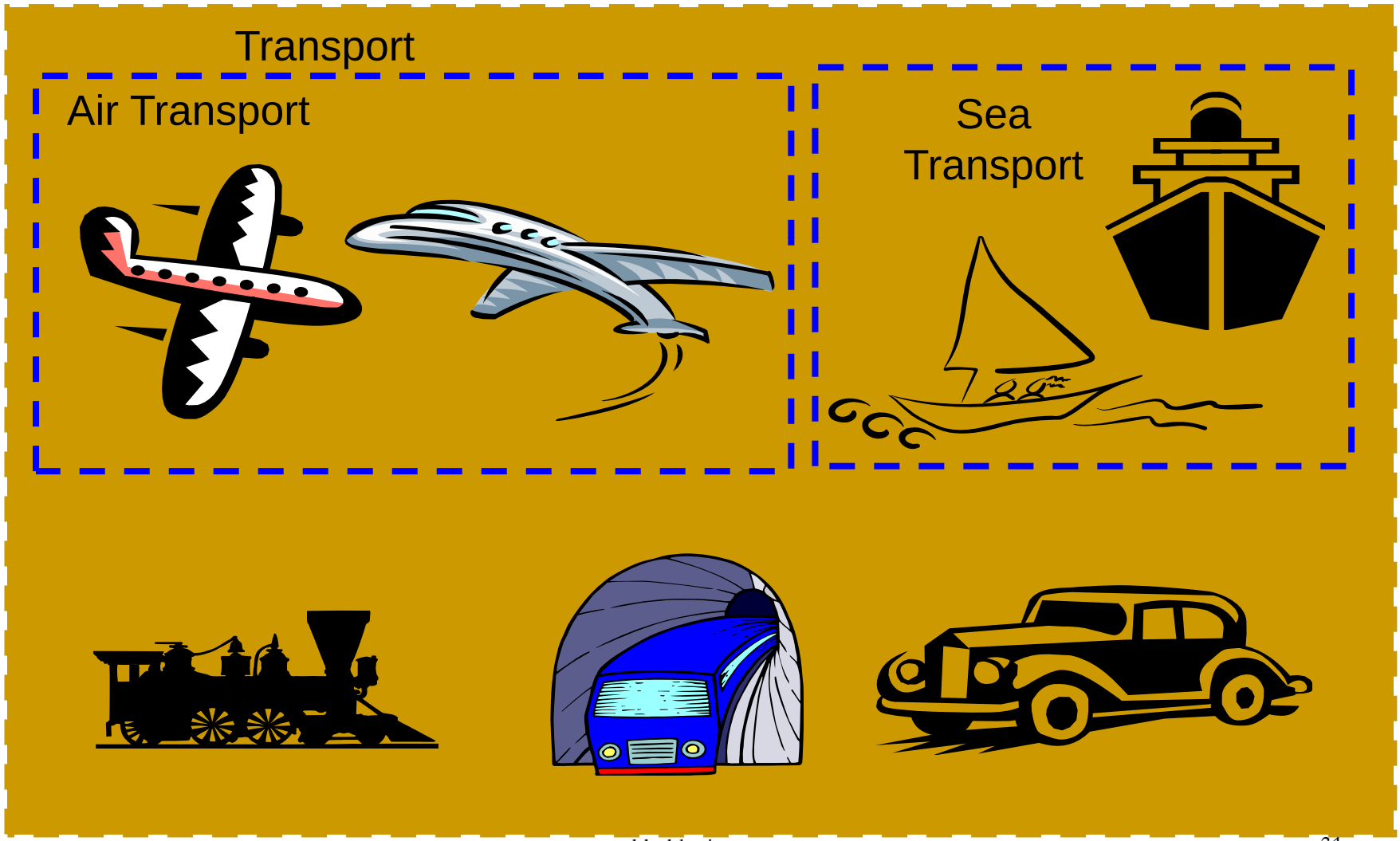
Inheritance Relationship



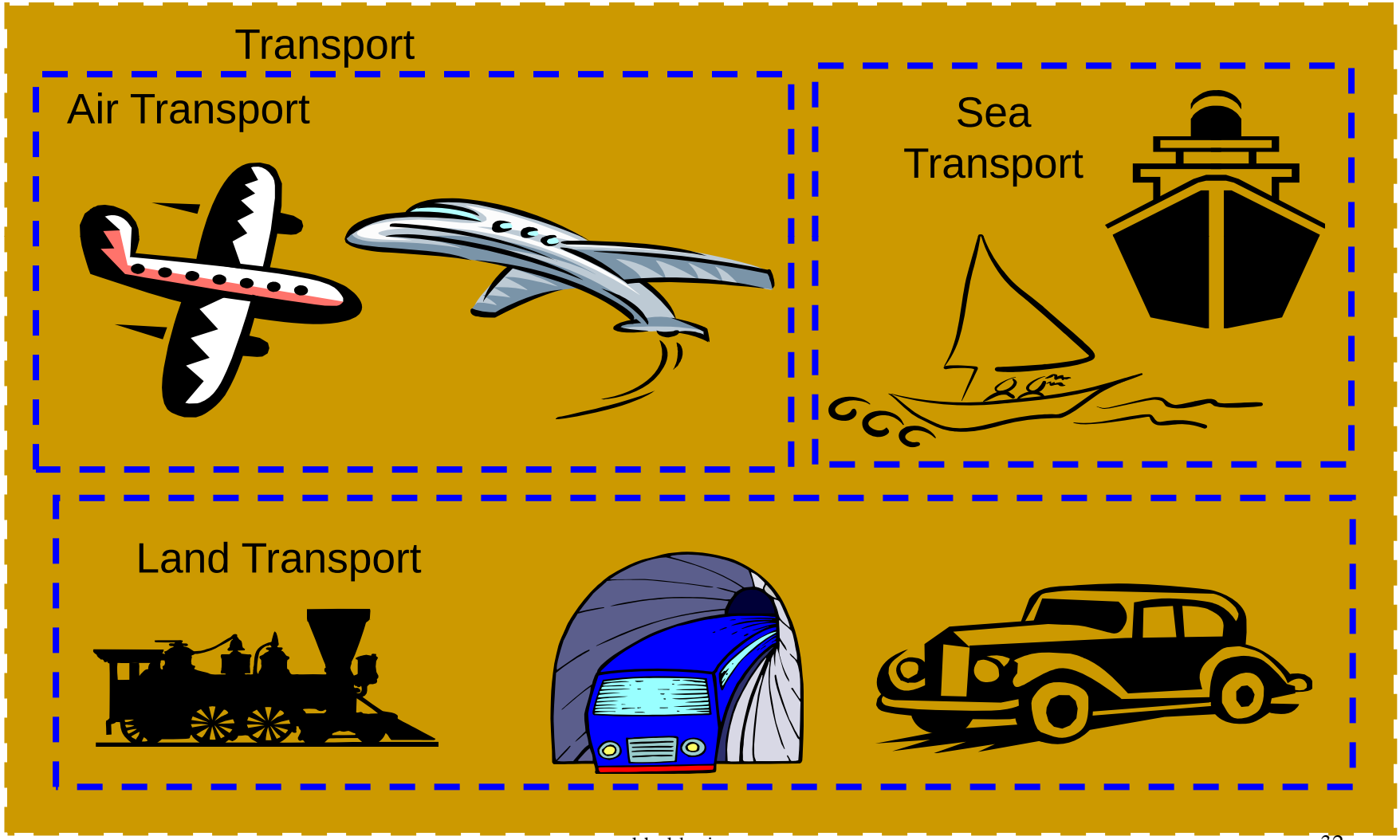
Inheritance Relationship



Inheritance Relationship



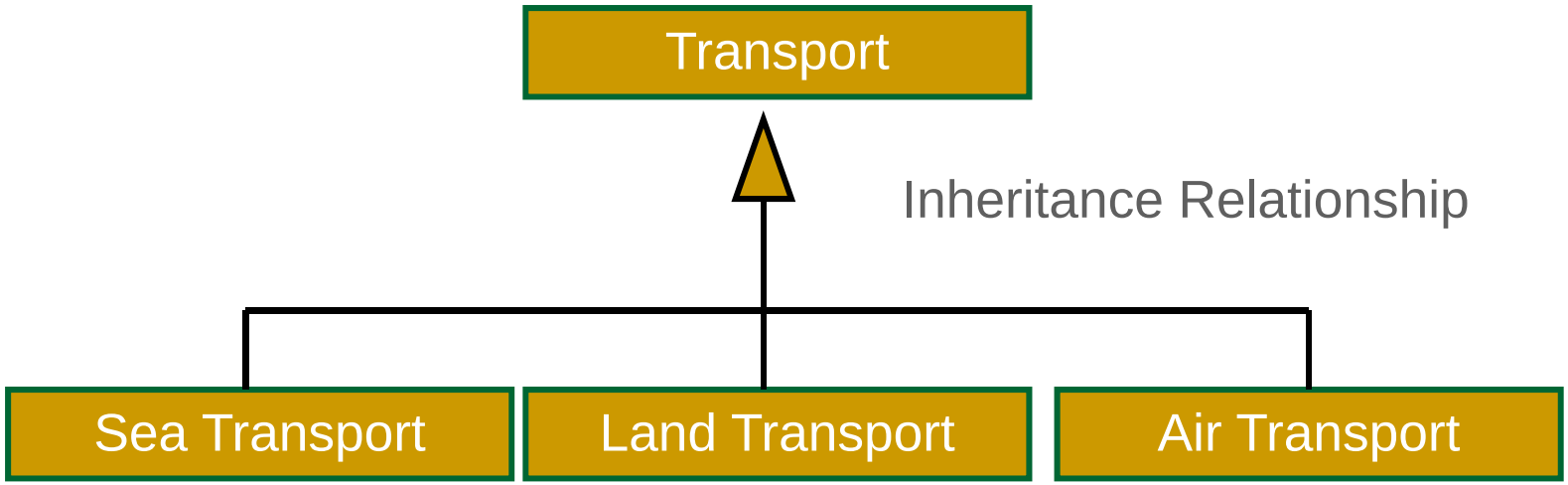
Inheritance Relationship

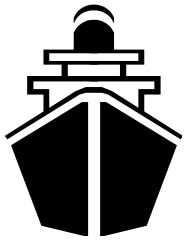


Inheritance Relationship

- The inheritance relationship is sometimes called the “is-a” relationship.
  - For example...

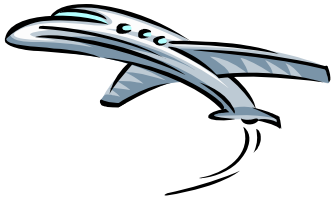
Inheritance Relationship





Sea Transport object "is a" Transport object

Land Transport object "is a" Transport object



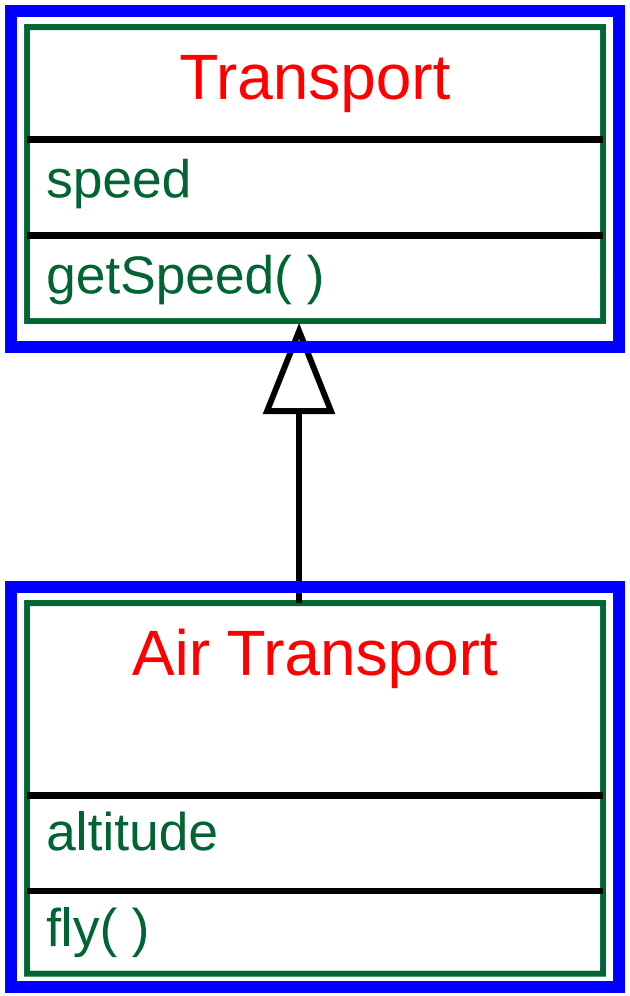
Air Transport object "is a" Transport object

Inheritance Relationship

- Based on the meaning of the “is-a” relationship, each object will have attributes and behaviour as defined by its category and all its supercategories.

- An inheritance hierarchy can be built through generalization or specialization.

Inheritance Relationship



Attributes

speed

altitude

Behaviour

getSpeed()

fly()

Questions

THANK YOU

Arash Habibi Lashkari

PHD. Candidate of UTM

Kuala Lumpur, Malaysia

Feb, 2010

THE END